UEF4.3. Programmation Orientée Objet

Examen Final

Durée : 02 heures

Documents interdits

Exercice 1 (10 pts)

On désire écrire un programme qui lit un fichier texte et crée une liste triée par ordre alphabétique de tous les mots contenus dans ce fichier. Cette liste de mots est ensuite stockée dans un autre fichier. Pour cela, tous les mots sont lus à partir du fichier d'entrée et stockés dans une structure de données suivant l'ordre alphabétique et sans répétition. Les mots contenus dans la structure de données sont ensuite écrits dans le fichier de sortie, un mot par ligne (Un mot dans ce programme est défini comme étant n'importe quelle chaine de caractères).

A/ Compléter le code suivant pour obtenir le programme voulu. Prévoir la gestion des erreurs :

```
// Importation des packages
.....
public class ListeMots{
// 1. Déclaration des attributs
.....
/* Programme principal*/
public static void main (String args[]){
// 2. Déclaration des variables
.....
/*3. Créer un flux d'entrée */
.....
/*4. Créer un flux de sortie*/
.....
/* 5. Lire tous les mots à partir du flux d'entrée et les stocker dans la structure de donnée adéquate. Pour cela, utiliser la méthde insererMot que vous définirez plus bas*/
.....
/* 6. Ecrire tous les mots à partir de la structure de données dans le fichier de sortie*/
}// Fin du programme principal
```

UEF4.3. Programmation Orientée Objet

```
/* 7. La méthode insérer mot. Elle reçoit un mot en paramètre et l'insère dans la structure de
donnée déclarée dans le main() */

void insererMot (String m) {

.....

}// fin de la méthode insererMot

}// Fin de la classe listeMots
```

Remarque : Pour insérer une nouvelle ligne, utilisez la méthode `newLine()` du flux de sortie.

B/ Nous souhaitons faire ce traitement à un ensemble de fichiers. A chaque exécution du programme, un fichier sera traité, et lorsque tous les fichiers auront été traités, nous obtenons dans le fichier de sortie tous les mots apparaissant dans l'ensemble des fichiers, triés par ordre alphabétique. Quel est le mécanisme orienté objet permettant de le faire ?

C/ Nous voulons maintenant modifier le programme afin qu'il garde trace du nombre d'occurrences de chaque mot du fichier d'entrée. Le programme crée un fichier où les mots sont triés selon l'ordre décroissant du nombre d'apparitions. Pour cela, nous utilisons une classe appelée InfoMot qui est instanciée à chaque fois qu'un mot est rencontré pour la première fois.

```
class InfoMot {
      String mot;
      int nbOcc;
      public infoMot(String m) {
            Mot = m;
            nbOcc = 1;
      }
}
```

Adaptez la classe InfoMot ainsi que le programme écrit dans la partie A, sans modifier le «`main`» afin de répondre à ce nouveau besoin. Expliquez en quelques points et donnez uniquement le code modifié en java.

D/ Nous souhaitons maintenant créer un fichier index d'un document fourni en entrée précisant pour chaque mot le numéro de la ligne ou des lignes où il apparaît. Quelles est la structure de donnée nécessaire ? Expliquez.

UEF4.3. Programmation Orientée Objet

Exercice 2 (10 pts)

Dans un système d'exploitation, en plus des processus système, à chaque lancement d'un programme, un processus est rajouté à la liste des processus actifs. Nous supposons que si l'utilisateur lance deux fois le programme, deux instances du processus sont insérées. Dès que l'utilisateur quitte le programme, son processus est retiré. Cette opération n'est possible que si toutes les fenêtres associées à ce programme sont fermées. Evidement, les processus système ne peuvent pas être retirés.

1. Quelles sont les classes nécessaires pour représenter cette gestion de processus ?
2. Parmi ces classes, l'une utilise un attribut pour représenter les processus actifs. Voici une partie du squelette de la classe :

```
public class ….{
private ….. processusActifs = new ……. ;
public void retirer (……) {
……
    }
……
}
```

   a. Parmi les classes identifiées dans la question 1, quelle est la classe correspondante ?
   b. Quelle structure de donnée proposez-vous pour gérer les processus actifs (attribut « processusActifs »)?
   c. Implémentez en Java la méthode « retirer » de cette classe qui permet de retirer un processus actif. Préciser si nécessaire, les attributs (avec leurs types) et/ou les méthodes (leurs entêtes) à rajouter dans cette classe ou dans d'autres classes.
   d. Nous voulons que la méthode « retirer » lance une erreur de type RetraitErreur si le retrait est impossible. Modifiez le code précédent et implémentez les mécanismes permettant de le faire.

3. Nous souhaitons maintenant offrir la possibilité de trier les processus par rapport à l'espace occupé en mémoire. Quelles sont les modifications nécessaires pour pouvoir le faire ? Expliquez en quelques points et donnez les fragments de code nécessaires en java.
4. Si nous voulons changer la politique d'insertion des processus en politique LIFO, de quel type deviendra l'attribut « processusActifs » de la question 2.b ? Justifiez sans donner le code.